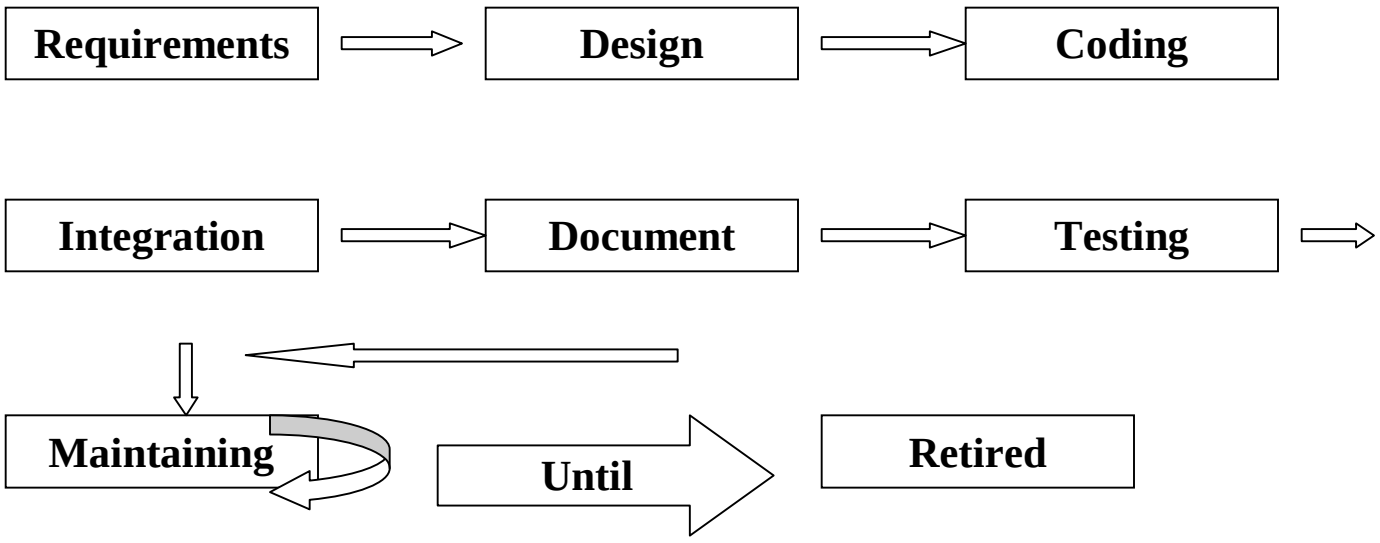
Software Engineering
PART 1

ان معنى كلمة Software تشير الى البرامج والاجرائات وكل ما يتعلق بها من وثائق تبين طريقة عملها وتخطيطها وسرد جميع تفاصيلها كخرائط التدفق التى توضح كيف صنعت البرامج وكيفية استخدامها

ومن هذا التعريف فكلمة Software لاتعنى برنامج وحسب بل تعنى كل ما يتعلق بهذا البرنامج وكافة التفاصيل

دورة حياة النظام او Software Life Cycle :

والمقصود بها جميع مراحل النظام من بداية التخطيط له مرورا بتصميم اجزائه الى مرحلة استخدام لغات البرمجه معه وفى النهايه اجراء الاختبارات لتحديد قدراته وكفائته



وتنتهى دورة حياة النظام عندما يتقاعد او يصل الى مرحلة يلزم فيها تطويره او تغيره

وتستمر مرحلة الصيانه حسب متطلبات المستخدمين طول فترة حياة النظام

مع ملاحظة انه يجب توثيق كافة هذه المراحل ورفع تقارير مستمره لمصممين النظام

أنواع Software : هناك عدة انواع للانظمه التى يتم صنعها ومنها

System Software: وتتعلق بأنظمة التشغيل التى تعمل عليها مختلف الاجهزه

الاجهزه الاليكترونيه ومن امثلتها

Operating System – Utilities – Compilers – Debuggers - Assembles

وهناك البرامج المنفرده والتى لها تصنيفات عديده داخل لغات البرمجه وتنقسم الى نوعين

رئيسين هما :

Generic Software

ويصمم لقطاع عريض من المستخدمين كالتى تستخدمها مع اجهزة الشخصيه من برامج

وسائط متعدده كالصوت والفيديو او برامج الرسوميه او برامجك المكتبيه

Customized Software

تصمم لمستخدم بعينه اى برامج متفرده فى استخدامها كبرامج الشركات والحسابات

وادارة الالات والتى لا تصلح للاستخدام العام

- لكن على اى حال تستطيع بسهوله التفريق بين التصنفين السابقين على اساس واحد

1 – اذا كان مطور النظام او Developer هو الذى يتحكم فى ادارة النظام اذا فهو يصلح للاستخدام لقاعده عريضه من المستخدمين اى Generic

2 – اذا كان المستخدم هو الذى يتحكم فى ادارة النظام اذا فهو صنع خصيصا له اى Customized

تحديات النظام او Software Crisis :

1 – دائما ما يخرج انتاج النظام عن الجدول الزمنى المحدد له وتكون التكلفه اعلى من الميزانيه الموضوعه له

2 – تكون الكفائه اقل من المتوقع احيانا

3 – وسائل الحوار بين المستخدمين والمطورين ضعيفه وغير كافيه لاستيفاء كافة متطلبات النظام

وكافة هذه التحديات ما زالت موجوده ويتم ابتاكار اساليب جديده للتغلب عليها

هندسة البرمجيات ومراحل التحول فى بناء الانظمه
Software Engineering and Evolution of Software

فى البدايه كانت عملية التطوير تعتمد على المجهود الذاتى دون وضع اى معايير او ضوابط او جدول زمنى محدد الى ان تم ابتكار اسلوب هندسة البرمجيات وهو علم يختص بانتاج الانظمه المختلفه مهما اختلفت تصنيفاتها او حتى اللغه المستخدمه فيها

نماذج تخطيط مراحل النظام او Software Development Life Cycle Model :

SDL وهى الخطه التى تسير عليها لانهاء مراحل النظام كما انها تسهل عليك تطوير النظام لكن هناك اساسيات لا يتم الاتسغناء عنها ايا كانت الطريقه المستخدمه وهما :

الكفائه – الانتاجيه Quality and productivity

وتتاثر هذه الاساسيات بعدة عوامل وهى :

1 – كفائة المبرمجين people Quality

2 – الخطط التى وضعت Process Model

3 – الهارد وير الذى سيعمل عليه النظام Tools and technology هل موجود هل رخيص هل سيتم استيراده

- لماذا استخدام SDL- Model :

1 – تساعد على فهم خطة العمل كاملة

2 – تساعد على بناء اقتراحات لبناء النظام

3 – تستطيع ان تتحكم فى مصادر النظام لاحقا

4 – تساعدك على مراقبة نجاح النظام اولا بأول عن طريق تجربة كل شيء بعد كل مرحله تم انجازاها

- دراسة الجدوى او Feasibility Study :

من المهم جدا عمل دراسة جدوى لمشاريعك حيث تستطيع تحديد الاتى بواسطتها :

1 – جميع مصادر النظام او Resources

2 – جميع المتطلبات او Requirements

3 – التكلفه الكليه للمشروع Costs

4 – المنافع التى تعود على المستخدمين لهذا النظام Benefits

ان دراسة الجدوى للمشاريع لها انواع حيث ان كل منها له استخدامه الخاص به

1 – Organization Feasibility :

ووظيفتها معرفة هل النظام يدعم المتطلبات التى يحتاج اليها المستخدمين مثلا

هل يدعم العمل على الشبكات – هل يستطيع العمل مع انظمه اخرى

2 – Economic Feasibility :

دراسة الجدوى الاقتصاديه هل النظام سيعود على الشركه بالمنافع هل سيزيد الانتاج بدون استيراد الالات اخرى هل العائد المادى اعلى من تكلفة المشروع وما يتعلق به من هاردوير

3 – Technical Feasibility :

هل الهاردوير الذى سيعمل عليه النظام او Software تستطيع الشركه ان تحصل عليه او استيراده

4 – Operational Feasibility :

هل الموردين قادرين على تمويل المشروع وهل المستخدمين قادرين على شرائه وهل هناك موظفين قادرين على التدرب عليه جيدا

صيانة النظام او Maintenance :

وتعتبر جزء هام جدا من مراحل النظام وتستهلك وقت ومجهود اكبر من انشاء النظام نفسه ولها تكاليف قد تصل الى 50% او 80% من تكلفة النظام وليس من الضرورى ان يوجد عيوب او اخطاء فى النظام لكى يتم صيانته ولها انواع

انواع الصيانه للنظام Types of Maintenance :

: Corrective Maintenance – 1

تعتبر هى الاقل تكلفه وتحدث نتيجة ان بعض العمليات او Process داخل النظام لاتعمل بشكل جيد او هناك تحميل زائد على النظام

: Adaptive Maintenance – 2

تحدث نتيجة تغيير فى بيئة عمل النظام وسوف يتيغير النظام لكى يعمل مع البيئه الجديده والمقصود بها انه حدث تغيير مثلا فى الهاردوير المستخدم او حدث تغيير فى المدخلات او المخرجات للنظام

: Preventive Maintenance – 3

تحسين جديد على النظام حتى يتم حمايته من الاهمال او التقاعد وذلك بسبب وجود تكنولوجيا جديده فى الاسواق وتسمى عملية الهندسه العكسيه وهى اعادة احياء النظام من جديد ولها نوعان :

: Black Box – 1

وينظر الى المدخلات والمخرجات للنظام القديم ثم يصنع خوارزميات تقوم بنفس العمل وتتكيف مع الهاردوير الجديد

: White Box – 2

يقوم بتغيير عمليات النظام من الداخل او تغيير Process

متطلبات مرحلة Requirements :

ان هذه المرحله لها شروط لكى يتم عملها بشكل صحيح وذلك لان النظام كله سوف يبنى عليها

1 – Requirements Documents :

ان تحتوى على خلاصة الاشياء التى يحتاجها المستخدم دون زياده او نقصان

2 – Contradiction :

ان لا تتعارض مع بعضها

3 – Precise :

ان تكون محدده

4 – Ambiguity :

عدم الغموض

5 – Complement :

متكامله

6 – Constant :

متماسكه

2 – مرحلة Project Plan :

الخطه التى تسير عليها لبناء النظام

4 – مرحلة Design :

ونتبع فيها لغه معينه فى التصميم وسيتم شرحها فيما بعد

5 – مرحلة Final Coding :

وفيها نستخدم لغة برمجه معينه لكتابة الكود

6 – مرحلة Test plan and Test reports :

مرحلة الاختبار وعمل تقارير توضح نتيجة الاختبارات على النظام

7 – مرحلة S.w Manuals :

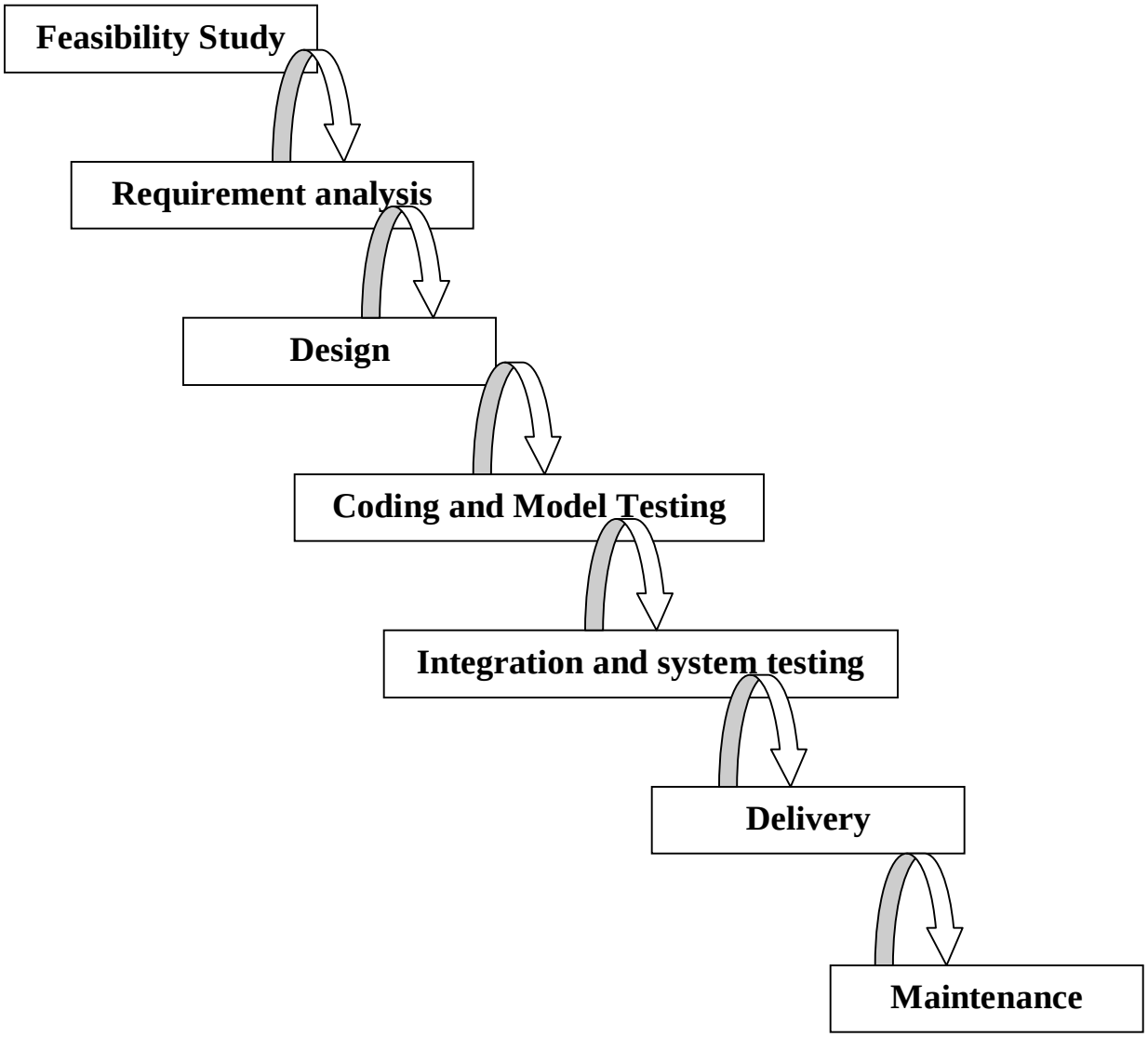
تدريب المستخدمين على عمل النظام

أنواع تخطيط العمليات او Process Models :

سنقوم بذكر بعضها وطريقة عملها وتعتبر شيء هام جدا يجب التدرب عليه جيدا لانه يوضح اجزاء النظام والتى يتكون منها ويجب ان تعمل مع بعضها لكى تنتج فى النهايه النظام النهائى . ان اى عطل فى اداء Process يتعطل النظام بالكامل لذلك يجب الحذر عند العمل عليها

1 – طريقة الشلالات Water Fall Model :

طريقة الشلالات فكرتها ان كل Phase او مرحله من مراحل النظام لا تحتاج الى العوده الى المرحله السابقه بعنى انه بعد الانتهاء من كل مرحله لا يتم العوده اليها مجددا الى ان ينتهى بناء النظام



مزايا طريقة الشلال :

1 – سهولة ادارة النظام بمراحله

2 – كل مرحله يتم الانتهاء منها من الممكن ان تعمل كمنتج

عيوب هذه الطريقه :

1 – تجمد Requirements بمعنى انه بمجرد الانتهاء من هذه المرحله فلن تعود اليها مرة اخرى وبالتالى اكتشاف اخطاء فيها سيكون متأخرا كما ان المستخدم لا يستطيع اضافة اشياء اخرى عليها

2 – Big Bang :

وهى ان يصطدم المستخدم بالتصميم مره واحده واما ان يوافق عليه او لا يوافق

3 – تضخم Requirements :

حيث يتم جمعها مره واحده

ولحل مشكلة تضخم وتجميد Requirements نأتى الى طريقه اخرى وهى طريقة

2 – طريقة Prototyping Process Model :

طريقة عملها ان يرسم التصميم بالاجتماع مع المستخدم بمعنى ان تأخذ منه المتطلبات وتقوم بعمل Design وعرضه عليه وبالتالى تم الاستغناء عن مرحلة Requirements التى كانت فى طريقة الشلال

المزايا : -

1 – تأخذ وقت قصير ومناسبه للمشاريع كبيرة الحجم

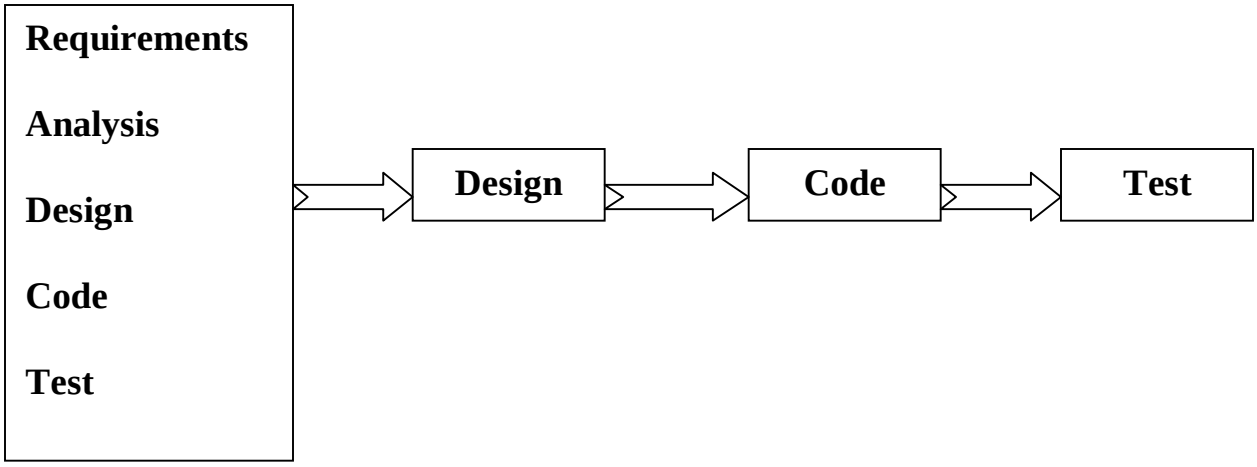
2 – عند عدم وجود Design يستطيع احتواء النظام كاملا

3 – تدريب المستخدم على الاستخدام اولا باول وتسمى عملية Deployment

العيوب : -

1 – تكلفه عاليه

شرط استخدام هذه الطريقه مهارات عاليه واستخدام اداوات عالية المستوى لتسهيل عملية التصميم



جدير بالذكر ان هناك طرق اخرى عديده سنذكرها للعلم فقط وهى

Iterative Enhancement – Time Boxing – Extreme Programming

- Requirements Engineering :

ان اساس عمل اى Software ناجح هو ان يكون Requirements Analysis له دقه عاليه ويلبى كافة المتطلبات ولذلك يجب التنويه عن انواعها :

1 – Functional Requirements :

وتدعى المتطلبات الوظيفيه والتى تصف بعض المعاملات الهامه للنظام كالمدخلات والمخرجات – البيانات التى سيتم تخزينها داخل النظام – الهيكل التركيبى للبيانات

: Non Functional Requirements – 2

المتطلبات الغير وظيفيه والتى تصف خصائص النظام – الكفائه التى يعمل بها – والاداء وعلى هذا فأن هناك بعض المصطلحات الهامه جدا داخل عالم Software والتى يجب التعرف عليها جيدا

Usability: - الاستخدام ويعنى هل مسموح بقاعده عريضه ام مخصص

Efficiency: - الكفائه كفائة النظام فى العمل

Performance: - الاداء بمعنى انه يؤدى أشياء محدده تحت Load او ضغط معين

Reliability: - يعتمد عليه جيدا

Portability: - قابل للحمل بمعنى العمل على انظمه مختلفه واجهزه مختلفه

Traceability: - الاحتفاظ بنسخه اصليه من كل عملية تعديل

تصنيفات Requirements :

: Volatile Requirements – 1

التى تتغير اثناء انشاء النظام اى انها ليست ثابته

: Enduring Requirements – 2

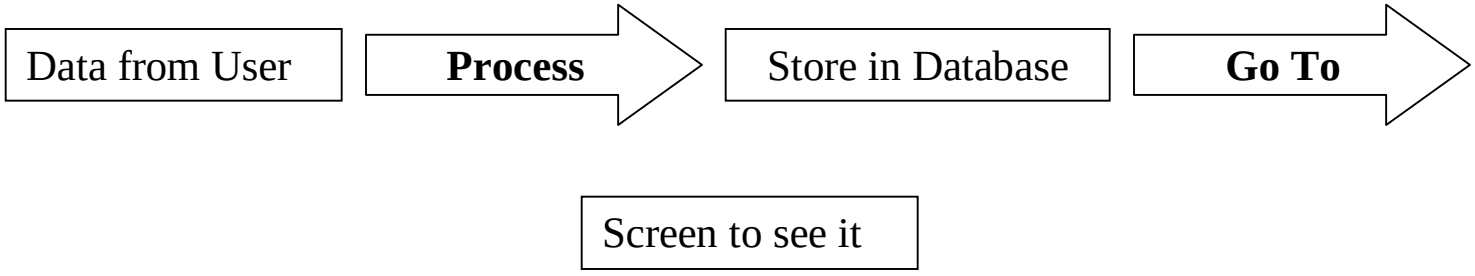
تظل ثابته لا تتغير

بعض انواع التصاميم التى نستعرضها اثناء مراحل بناء النظام :

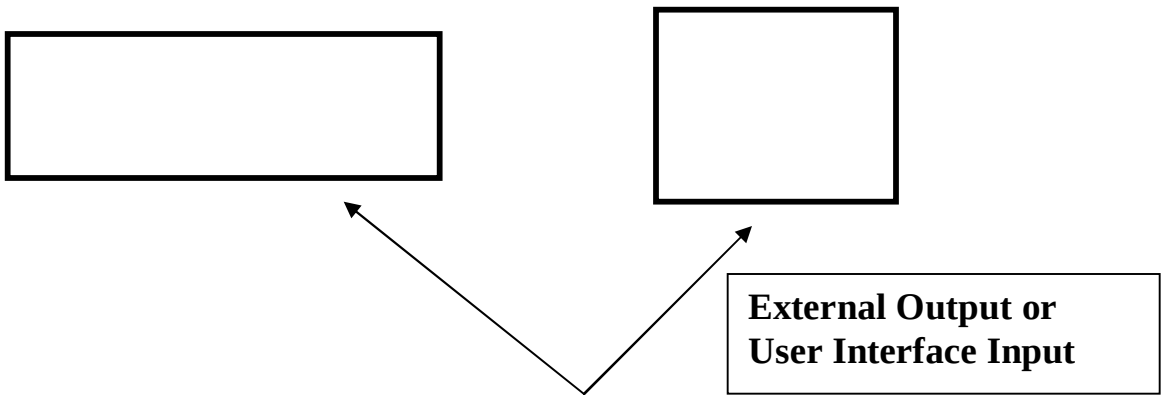
- :Data Flow Diagram DFD

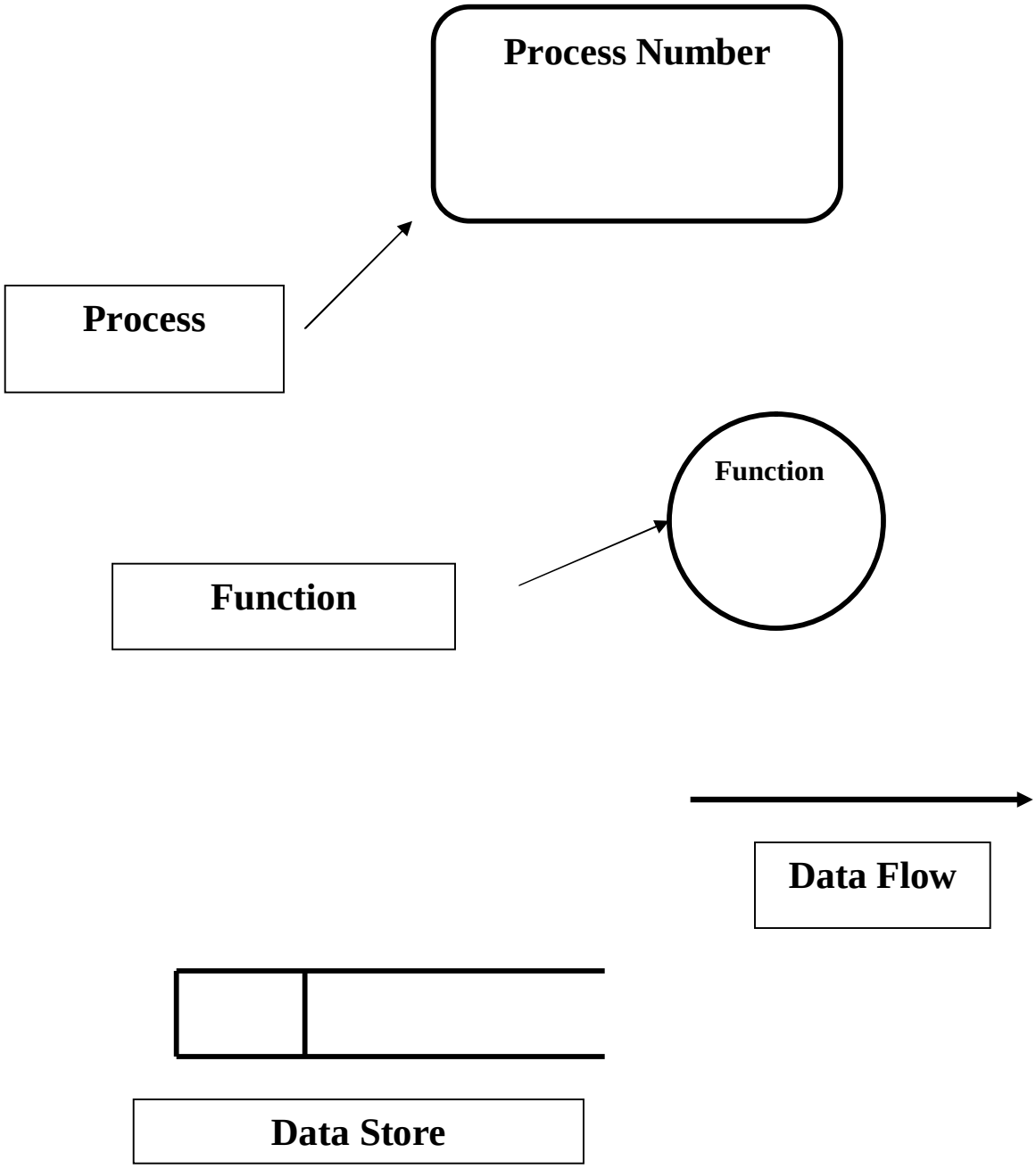
هو عباره عن تصميم يمثل النظام ككل بمعنى انه يوضح المدخلات والمخرجات

والعمليات التى تتم على مدخلات النظام لكى ينتج لنا المخرجات ببساطه



Some Important Symbols





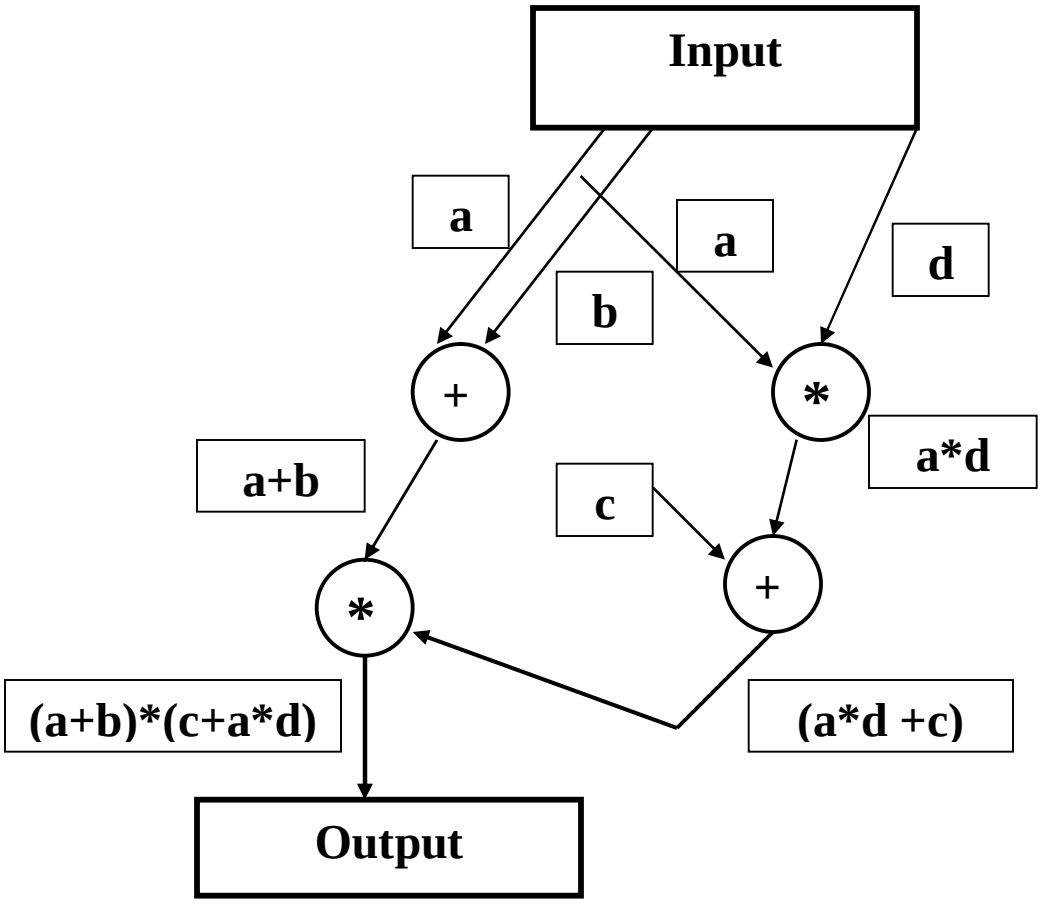
كانت هذه بعض رموز الهامه التى تصف مرور البيانات عبر اجزاء النظام

المثال التالى يوضح استخدام هذه الرموز

- نريد استخدام الرموز فى حل المعادله الاتيه والتى تمثل كنظام نريد رسم مخطط البيانات له

$$(a + b) * (c + a * d)$$

المطلوب فى هذا النظام قراءة البيانات من المدخلات ثم اجراء العمليات اللازمه ثم اظهار الناتج

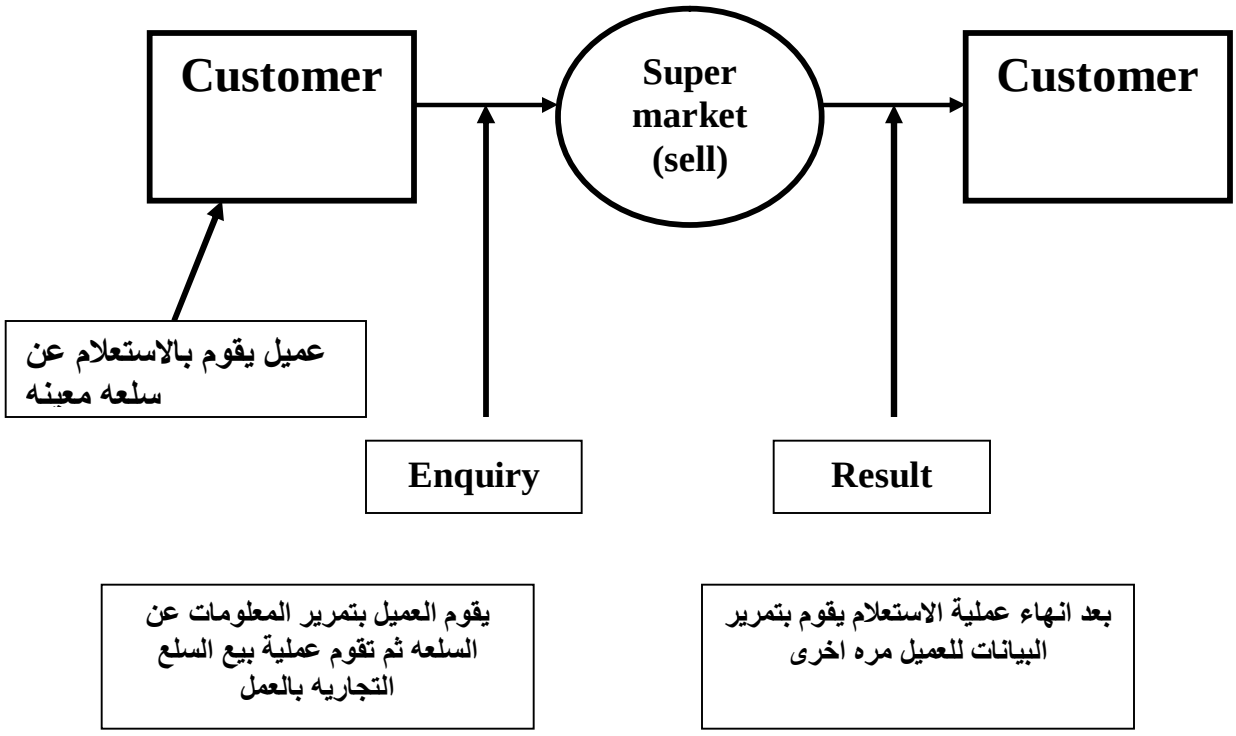


قد يكون تخطيط تدفق البيانات بسيطا لكنه من الممكن ان يأخذ عدة مراحل او بمعنى يتفكك الى مستويات عده ونرى ذلك فى الامثله الاتيه

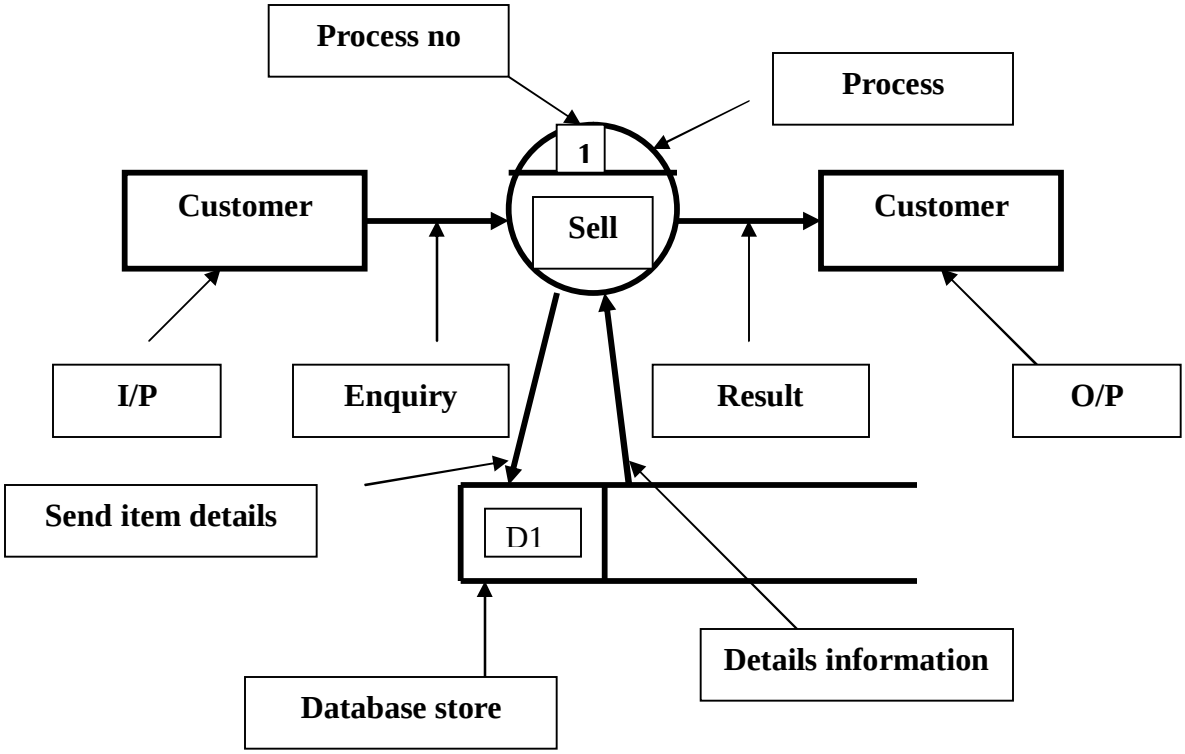
- Super Market Project : -

تخطيط لمشروع سوبر ماركت بواسطة تدفق البيانات حيث نرصد تحديد المدخلات والعمليات التى تحدث عليها الى ان نصل الى المخرجات وهنا يقسم الرسم الى عدة مستويات

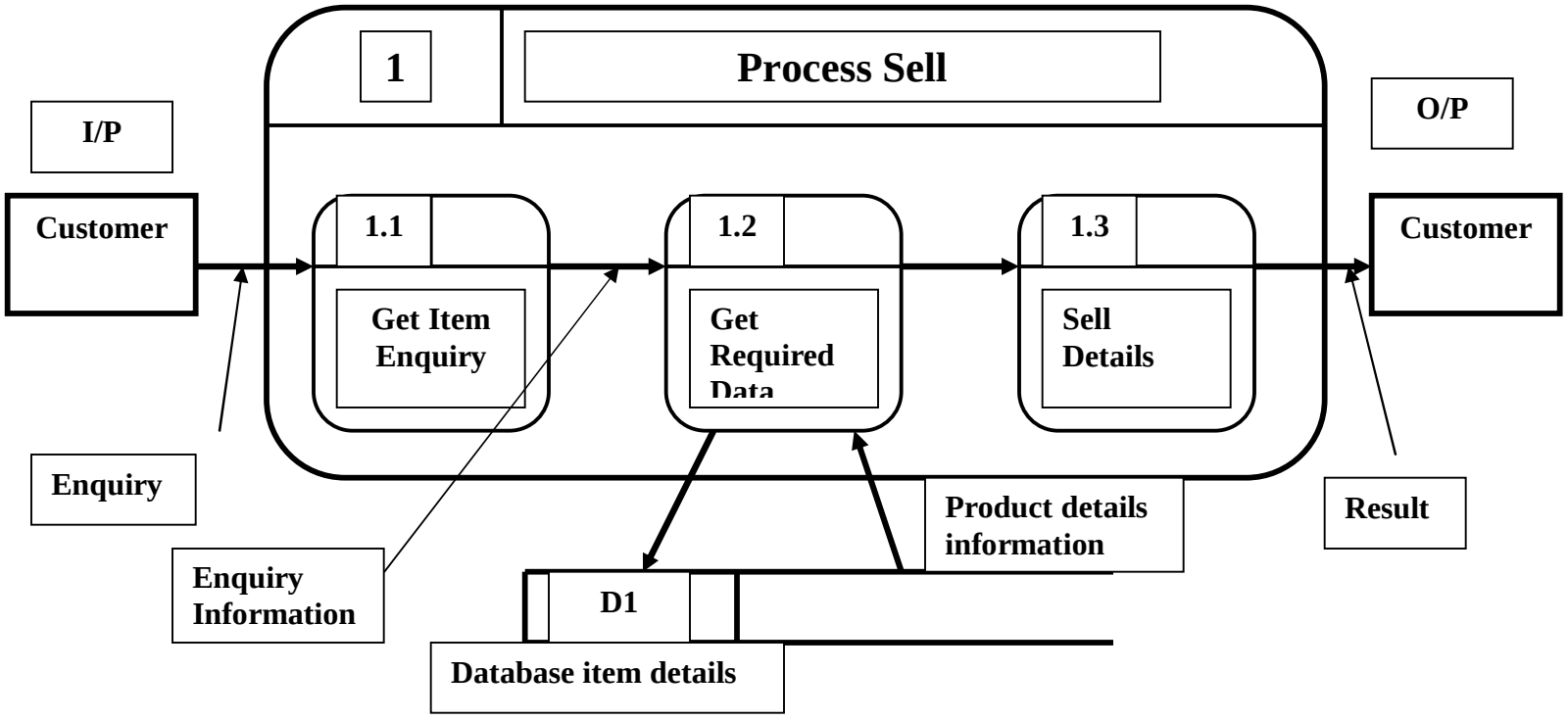
Level 0



Level 1

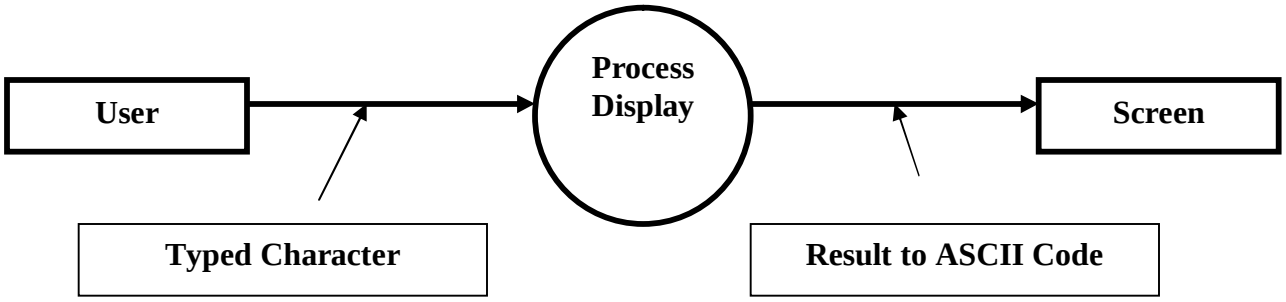


Level 2

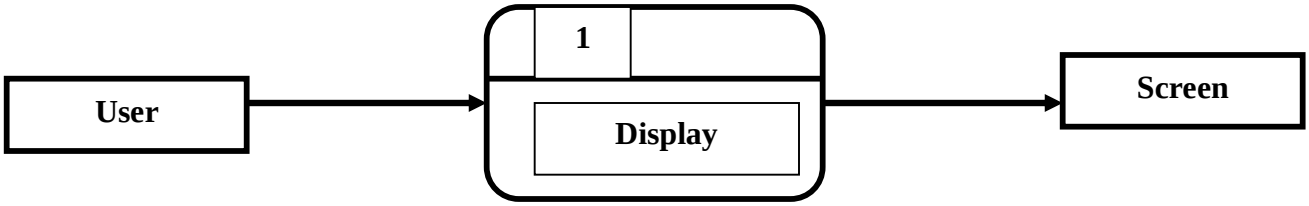


- مثال الصراف الالى ATM :

Level 0

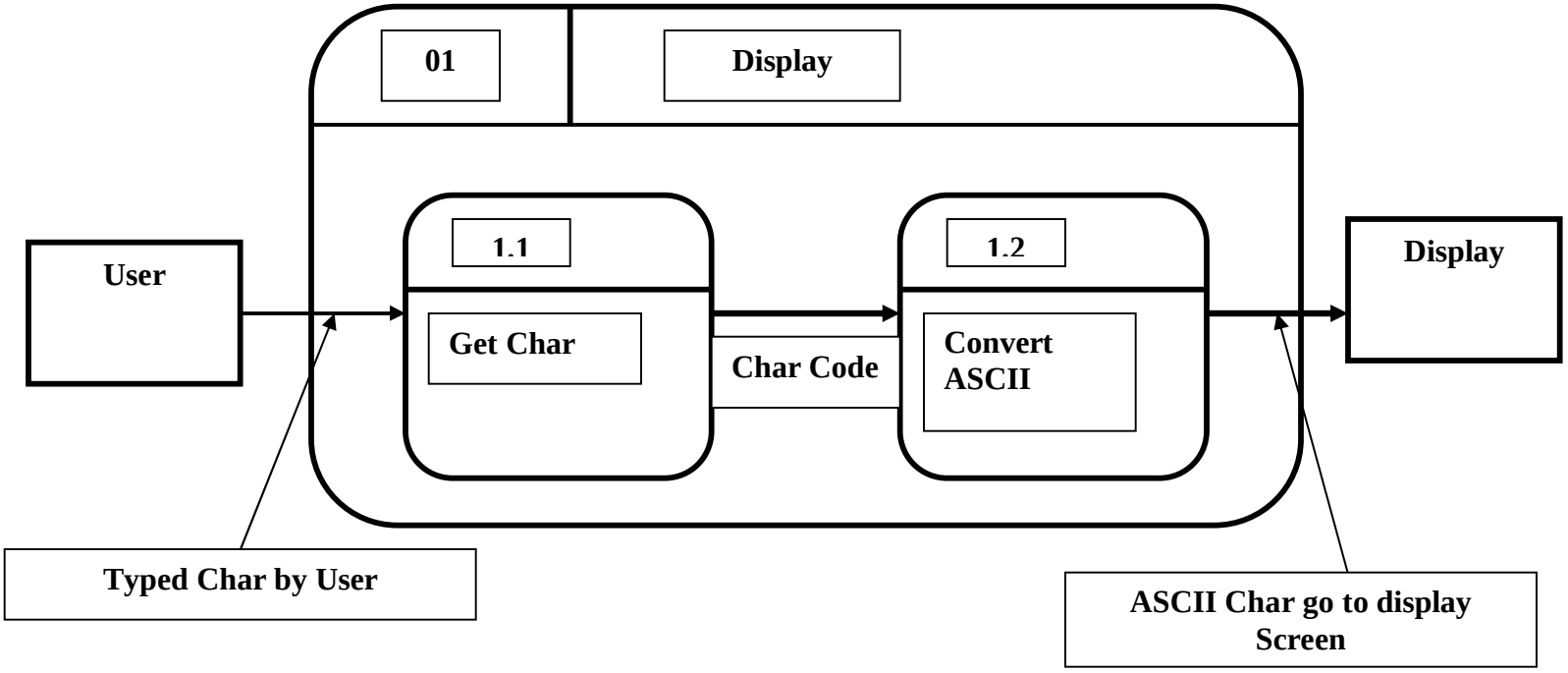


Level 1



Level 2

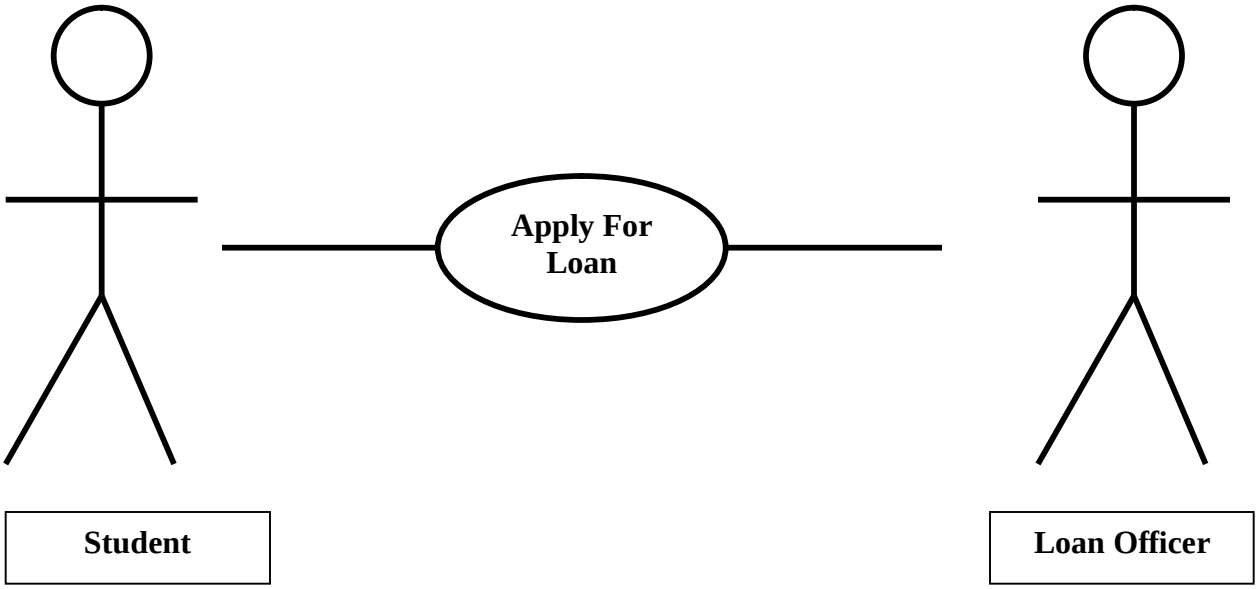
Level 2



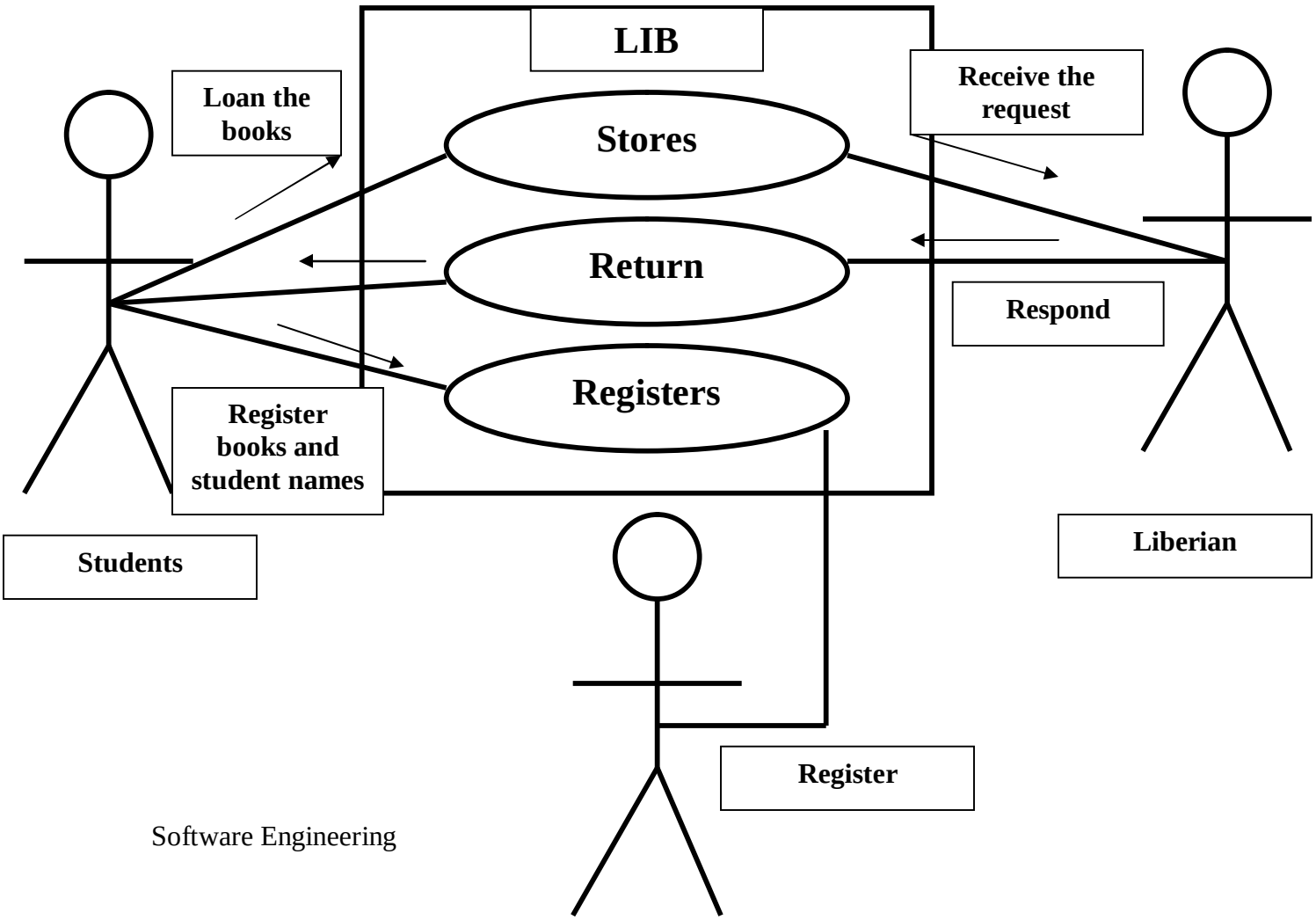
- ان بناء اى نظام تبدأ اولا بعملية حكايات عن النظام بمعنى وصفا النظام تفصيليا وتسمى ب Use Cases ثم بعد ذلك يتم عمل Diagram لها

ولدينا الامثله التاليه :

1 – مثال المكتبه : احد المكتبات التابعه للطلاب حيث يقوم الطلاب باستعارة هذه الكتب وبالتالى يتم اخذ بياناتهم ثم تقوم المكتبه باستعادة هذه الكتب مره اخرى بناءا على تلك البيانات كما يوفر النظام طريقة الاستعلامات



Relationship between Boss and Customers



1 – الطلاب سوف يقدومن طلبات باستعارة الكتب الى صاحب المكتبه

2 – يرد صاحب المكتبه بامكانية توفير هذه الكتب الان او انها غير متوفره ويقوم بالرد عليهم

3 – فى حالة ان الكتب موجوده يقوم المسجل او عامل الارشيف بتسجيل اسماء الطلاب الذى استعاروا بعض الكتب وبيانات الكتب

العمليه فى النهايه تعتمد على رؤيتك للسير النظام وبناءا عليه يتم تصميم كل شىء

الاشياء التى تدل عليها الرموز فى الشكل السابق :

- المستطيل يرمز الى النظام ككل
- الشكل البيضاوى يرمز الى حاله من حالات النظام او cases
- الخطوط الخارجه او الداخله الى النظام من ممثلين النظام تشير الى ان ممثل النظام مشارك فى هذه الحاله من حالات النظام
- الاسهم تشير الى اتجاه المعلومات داخل النظام

مفاهيم هامه :

1 – Primary Actor :

هو الممثل الريئسى للنظام وهو فى المثال السابق الطالب الذى صمم النظام من اجله

2 – Use Case :

عباره عن مجموعة من العمليات تنفذ عن طريق المستخدم وتستخدم فى النهايه عمل واحد فقط وتمثل بالشكل البيضاوى وتكتب فيه العمليه التى تنفذ

3 – الممثل او Actor : هو اى شىء يتعامل مع النظام من الداخل او الخارج وقد يكون المستخدم المباشر او شخص متأثر بالنظام او نظام اخر يتعامل مع النظام الحالى ويرمز له برمز الشخص او الفرد

4 – العلاقه او Relation : وهى ما يربط بين Use Case وبين Actors

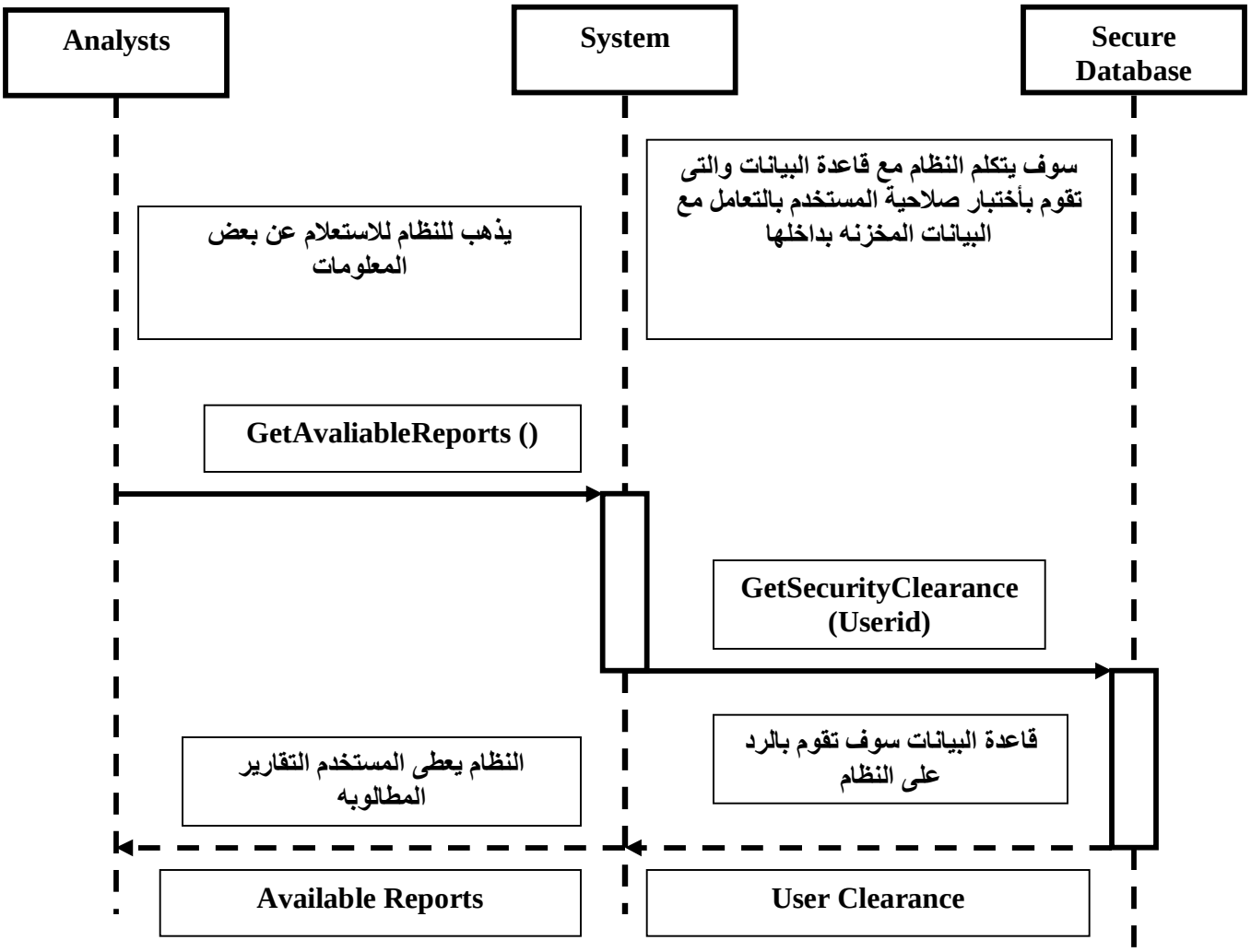
تستطيع استخدام طريقه اخرى قد تكون اكثر فهما وتوضيحا وتسمى

:Sequence Diagram

وهذه الطريقه هى الافضل فى تمثيل الخطوات الدقيقه للنظام وتعامل جميع اجزاء النظام

مع بعضها البعض مما يسمى Dynamic Behavior

لدينا مثال على ادارة نظام امنى يحمى قاعدة بيانات لمستخدمين معينين



الرسم السابق يصف جميع التفاصيل التى تحدث بين اجزاء النظام

1 – الخط المتقطع الراسى معناه ان سيقوم بعمل حدث معين داخل النظام او Process

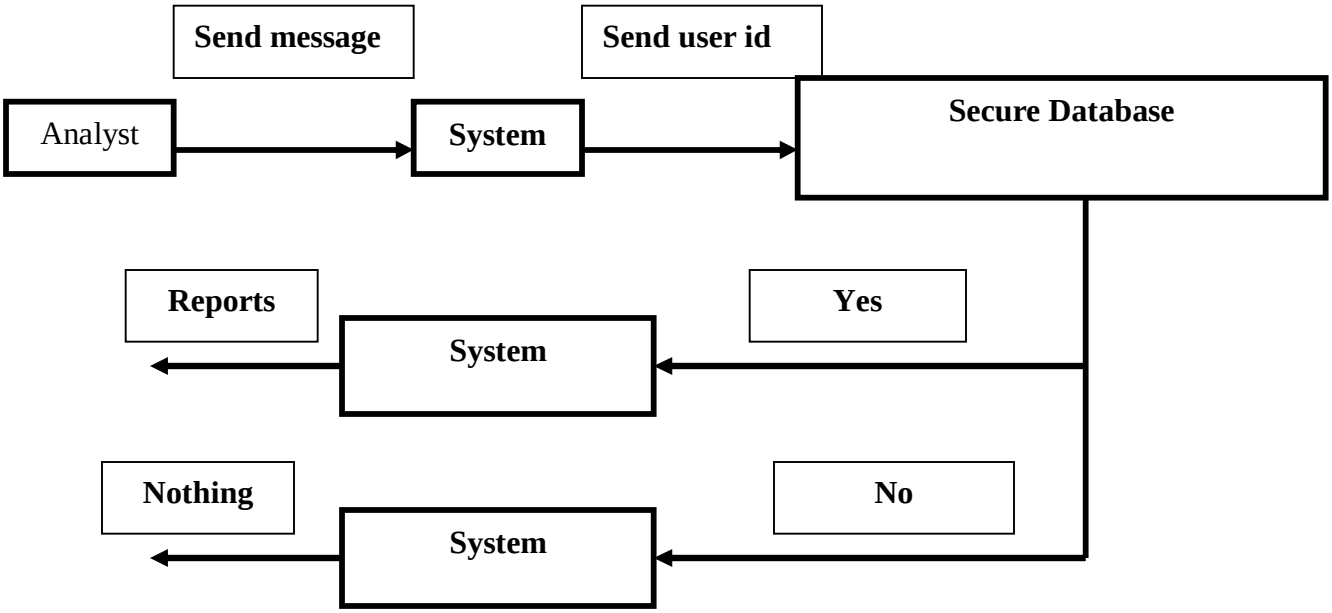
2 – السهم النقطى يسمى Life Line ويدل رسمه على الرد العائد من اجزاء النظام

3 – السهم العادى معناه استعلام او اوامر موجه الى جزء من اجزاء النظام

يبدء اعطاء امر الاستعلام للنظام بأول سهم فى Life Line يبدء من اليسار الى اليمين

وباقى الاسهم لابد ان تكون اسفل منه من والى اجزاء النظام

تستطيع عمل رسم كروكى توضيحى يوضح احتمالات النظام



فى النهايه تجد ان Sequence Diagram يربط بين كل خطوط سير النظام ليكون لديك

رؤيه عامه وكامله على سير النظام

System Requirements Specification:

SRS Document:

ويعتبر عنصر غايه فى الاهميه بالنسبه لعلم هندسة البرمجيات وله شكل ومنظور عالمى متفق عليه بين الاشخاص والهيئات التى تتعامل مع صناعة البرامجيات بصفه عامه ولذلك يعطونه اولويه بالنسبه لبناء مشاريعهم الضخمه ونأتى الان لتعريفه :

- اولا SRS لايحتوى على تصميم للمشروع ولكن هو وصف نصى كامل للنظام ولابد ان يكون Customer عنصرا اساسيا فيه لانه يعتبر مرجعية مصممين النظام والمكلفين بعمل SRS له واى خلاف فى النظام بين Customer وبين المصممين يتم الرجوع SRS ولهذا عند الانتهاء منه لابد من التصديق عليه من قبل المستخدم

- فوائده :

1 – يشكل القاعده للوصول الى اتفاق بين الشركه والعميل

2 – اذا تم بناء SRS بشكل صحيح سيعيفيك من اعادة التصميم والذى يبنى عليه

3 – تحديد بالظبط تكاليف المشروع Estimating Costs

4 – مصادقة العميل عليه Validation

5 – تامين على مصادقة العميل Verification

6 – سهولة استخدام SRS من نظام لاخر اذا ما تشابهت Requirements لهما او Facilitate Transfer

7 – يعطى رؤيه مستقبليه للمشروع Serves Enhancement

ويأتى فى هذا الاطار الصرح العملاق المسمى IEEE والمسئوله عن تطوير SRS للمشاريع العالميه وتعطى Standard لاى شيء مصنوع بمواصفات عالميه ومعترف به

محتويات SRS :

ما الذى يحتويه هذا الجزء الهام من مراحل النظام :

1 – Introduction او المقدمه

2 – Overall Description وصف عام ل SRS

3 – Specification Requirements متطلبات النظام

4 – Appendices الملحقات

5 – Index الفهارس

ونأتى الان لتفاصيل المقدمه :

يتكون هذه الجزء من SRS من عدة اساسيات وكل منهم له دوره الخاص فى توضيح النظام وهى كالتالى :

1 – Purpose الاهداف وهو تصوير بدقه الغرض من SRS وليس النظام ولمن صمم هذا SRS وتحديد الجهود المستهدفه

2 – Scope المجال ومعناه النطاق اى تحديد اسم معين للنظام والنطاق الذى يعمل به حتى اذا كان هناك اكثر من نطاق فلا تختلط الامور ببعضها

3 – Overview كيف نظم SRS وعلى ماذا يحتوى

4 – References المراجع اللازمه والتى تم الاعتماد عليها فى بناء SRS

5 – من المهم عمل الاتى بالنسبه للمشروع

- Definitions تعريفات لاختصارات قد يحتوى عليها SRS
- Acronyms وهو اختصار ينتج عنه كلمه جديده
- Abbreviations اختصار ولكن لاينتج عنه كلمه جديده

وبعد الانتهاء من تفاصيل المقدمه ناتى الى الاتى :

2 – Product Perspective هل النظام يعمل بذاته ام يعتمد على اشياء اخرى ومعنى هذا انه اذا كان نظامك جزء من نظام اكبر فيجب ان تذكر هذا الامر

3 – The Function ينبغى ان تنظم العمليه على شكل وظائف للنظام بشكل يفهمه المستخدم والمقصود به هنا ليس مستخدم النظام ولكن الذى سيقوم بقراءة SRS

<u>تفاصيل جزء Overall Description :</u>

1 – Regularly Policies او السياسات المنظمه لادراة المشروع

2 – Hardware Limitation ينبغى على Software الا يتخطى قدرات Hardware

توضيح تفاصيل SRS Specific Requirements :

1 – Functional Requirements اللوازم المتعلقه بالعمليات التى تحدث

2 – Performance الكفائه بمعنى ان يستجيب للعمل تحت ضغط معين ويوجد لها نوعان

Static لا يوجد وقت محدد لها

Dynamic تحافظ على Response Time

3 – Interface to other Applications ان يكون النظام له القدره على اتصال بأنظمه اخرى تعمل معه فى نفس بيئة العمل

4 – Audit Functions الدوال الدقيقه التى تستطيع ان يبنى عليها المبرمج قاعده له فى كتابة الكود او عملية Implementation

5 – Safety and Security Consideration ولها معنيان

Safety بمعنى تأمين النظام من الاستخدام الخاطئ او الغير منطقى من المستخدم

Security تأمين النظام من الاعمال التخريبيه المتعمده كالهاكرز او القرصنه

شروط SRS الجيد :

1 – Correctness ان يكون صحيح

2 – Completeness ان يكون كاملا

3 – Unambiguous ان لا يكون غامضا

4 – Verifiable ان يكون قابلا للتصميم

5 – Modifiable قابل للتعديل

6 – Traceable سهل فى عملية التتبع

7 – Consistency متماسك

8 – Testability قابل للاختبار

9 – Clarity واضح

10 – Feasibility ان يكون ذات جدوى

أنتهى الجزء الاول وسوف نقوم بتوضيح مثال عملى كامل فى الجزء الثانى على بناء SRS كامل لمشروع Security System كما سنقوم بسرد مراحل تصميم النظام وما هى الادوات المستخدمه فيه

memorycode_84@yahoo.com